[19]中华人民共和国专利局

[51]Int.Cl[6]
G06F 17/30

[12] 发明专利申请公开说明书

[21] 申请号 95191540.1

[43]公开日 1997 年 1 月 15 日

[11] 公开号 CN 1140500A

[22]申请日 95.2.6

[30]优先权

[32]94.2.8 [33]SE[31]9400410-8

[86]国际申请 PCT/SE95/00118 95.2.6

[87]国际公布 WO95/22111 英 95.8.17

[85]进入国家阶段日期 96.8.8

[71]申请人 艾利森电话股份有限公司

地址 瑞典斯德哥尔摩

[72]发明人 B·M·塞缪尔森

A·比扬纳斯泰特

[74]专利代理机构 中国专利代理(香港)有限公司

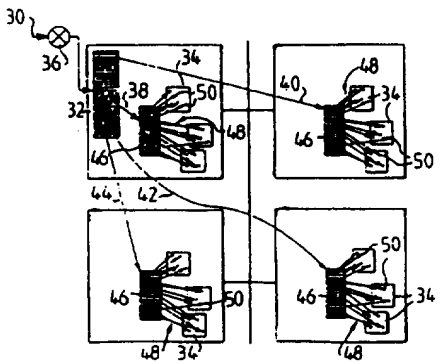
代理人 董 巍 王 岳

权利要求书 3 页 说明书 16 页 附图页数 7 页

[54]发明名称 分布式数据库系统

[57]摘要

在分布数据库系统中，数据库的不同部分由一些相互连接的处理器进行处理。数据库不同部分包括一些数据实体（58，66，66″,93），为每个这样的数据实体提供有关该数据实体所在处理器的全局信息（82），和关于该实体在本处理器中所处位置的本地信息（79，80，84），全局信息（82）在系统中的每个处理器中，以先前已定义的每个数据实体组共用及专用的形式存在。所述数据实体分布实体，每个分布实体又包括关于位于某个处理器中的某类数据实体实例组的信息，以便可以用来找到处理器地址的信息。



权　利　要　求　书

1.一个分布式数据库系统，包括一个数据库和用来控制处理所述数据库不同部分的一些互连处理器，该数据库部分包括一些数据实体，每个数据实体具有与之相关的信息，包括关于数据实体位于哪个处理器的全局信息，和数据实体位于本处理器何处的本地信息，该全局信息以通用全局信息的形式位于每一个系统中的处理器上，并且对每个预定义的数据实体集是特定的。

2.根据权利要求 1 的系统，其特征在于，上述数据实体集包含分布实体，每个分布实体包含关于位于某一处理器的数据实体具体类型的一些实例的信息，以及用于找到该处理器地址的信息。

3.根据权利要求 2 的系统，其特征在于，数据实体既可以用键值来寻址，也可以通过数据实体标识来寻址，数据实体标识除了含标识数据实体的信息外，还包含数据实体属于哪个分布实体的信息。

4.根据权利要求 3 的系统，其特征在于，数据实体标识包括本地和全局数据实体标识，每个本地数据标识包括两个信息域，第一个域标识本处理器，第二个信息域标识数据实体，每个全局数据实体包括两个信息域，第一个信息域标识分布实体，第二个域标识数据实体。

5.根据权利要求 4 的系统，其特征在于，所述本地信息至少包含在三张表中，这三张表是：第一张表包含本地数据实体标识，第二张表包含全局数据实体标识，至少有第三张表包含键值，所述全局信息包含在第四张表中，它包括分布实体编号，每个分布实体号指向另一个处理器。

6.在分布式数据库系统中，它包括一个数据库和用来控制处理所述数据库不同部分的一些互连的处理器，该数据库部分包括一些数据实体，每个数据实体具有相应的信息，包括有关数据实体位于哪个处理器的全局信息，和数据实体位于本处理器何处的本地信息，

一个利用数据实体唯一键值和类的标识号来访问属于某一特定类的数据实体的方法，包括以下步骤：

首先在本处理器，利用键值开始搜索数据实体，然后，

如果该搜索表明数据实体不在同一处理器，

产生一个逻辑分布实体号，它标识有关某处理器上数据实体类的一些实例的信息，以及这个处理器的寻址信息。

通过合成有关数据实体类的信息和逻辑分布实体编号，产生一个相应的物理分布实体编号，

利用物理分布实体号，标识在数据库中被搜索数据实体所在的处理器，

将消息发送给处理器，该处理器包含有关被搜索数据实体的信息，

利用键值，在已找到的处理器中，对要访问的数据实体作本地搜索，

将找到的数据实体的拷贝返回给已经请求访问的处理器。

7.在分布式数据库系统中，它包括一个数据库和用来控制处理所述数据库不同部分的一些互连处理器，所述数据库部分包括一些数据实体，每个数据实体具有相应的信息，包括有关数据实体位于哪个处理器的全局信息，和数据实体位于本处理器何处的本地信息，

一个访问属于数据实体特定类的数据实体的方法，包括以下步骤:

产生一个本地数据实体标识，它包含有关于分布实体的信息，分布实体除了包含标识数据实体的信息，还包含位于本处理器的数据实体类的一些实例的信息，

为了找到数据实体，在本处理器利用本地数据实体标识开始搜索，并

如果搜索结果表明数据实体不在本处理器，

将本地数据实体标识转换成全局实体标识，全局实体标识包括如下信息，

关于分布实体的信息，该信息又包括，在某一处理器上的数据实体一个特定类的一些实例信息，属于上述类的将被访问的数据实体，以及关于对上述处理器寻址的信息，

标识将被访问数据实体的信息，

利用包括在全局数据实体标识中的分布实体来搜索被访问数据实体所在的处理器，

向该处理器发送消息，消息中包含全局数据实体标识，

通过全局数据实体标识在已被找到的处理器中搜索数据实体标识，

将找到的数据实体标识的拷贝返回给开始访问的处理器。

8.分布式数据库系统中的分布实体，在此分布实体中，数据库的不同部分分别由一些互连的处理器控制处理，不同的数据库部分包含一些数据实体，上述分布实体包含位于某一处理器上一种特殊类型的数据实

体的一些实例信息，以及有关这个处理器的地址信息。

9.根据权利要求 8 的分布实体，其特征在于，它构成了数据实体标识的一部分，而数据实体标识除了包含标识数据实体的信息外，还包含有关分布实体的信息。

10.分布式数据库系统中的分布实体，在其中数据库的不同部分分别由一些互连处理器控制处理，不同的数据库部分包含一些数据实体，上述分布实体包含共用信息，并且对于预定义数据实体集来讲是专有的。

11.根据权利要求 10 的分布实体，其特征在于，它构成了数据实体标识的一部分，数据实体标识包含有关分布实体的信息以及标识数据实体的信息。

12.分布式数据库系统中数据实体的标识实体，在其中数据库的不同部分分别由一些互连处理器控制处理，不同的数据库部分包含了一些数据实体，上述标识实体包含了

有关分布实体的信息，在分布实体中又包含了以下信息：有关位于某一处理器上一种特殊类型数据实体的一些实例信息，该数据实体属于哪种类型的信息，以及对处理器寻址的信息。

说　明　书

分布式数据库系统

技术领域

本发明一般涉及一个分布式数据库系统，在该系统中，数据库的不同部分分别由一些互连的处理器控制处理，不同的数据库部分包含了一定数量的数据实体。

本发明具体涉及到数据库分布，也就是说，在当前连接中，数据实体在数据库系统中是如何寻址和分布的。

这里的数据实体是指诸如面向对象系统中类型对象的实体。

现有技术

在面向对象分布式数据库系统中，所包括的每个处理器除了需要访问其它处理器数据库部分的对象外，还可能需要访问本数据库部分中的对象。因此对于每个对象，就有涉及对象所在的子网和处理器的信息，有关于在另一个执行诸如获取对象之类所需服务的处理器中的、代理的信息，以及关于在处理器内存中所述对象的确切位置分布信息。如果所有的这些信息在所有的处理器上皆可用，那就得有很大的地址表，并且当一个对象创建、删除和移动时，需要在整个数据库系统上对对象的地址作大量的更新。

EP 405，829 涉及一个电信系统，其中的软件是用对象形式的独立软件组件实现的。在“运行中系统”中，函数“运行中连接器”记录对象并存储指向对象数据的数据指针。源对象往运行中系统发送消息来与其它对象通信。这些消息包括目标对象响应方法的名称和标识。

运行中系统只支持单个处理器或对象组，而且，如果目标对象属于由运行中系统支持的对象组，系统通过方法标识和数据指针来调用目标对象。如果目标对象处于另一处理器，运行中系统便向其它处理器广播这一消息。在每个接收处理器中，运行中系统检查它的“连接器表”。是否有这一消息目标对象的符号名，如果找到，便依据消息中的响应方法标识和运行中连接器中的数据指针信息调用目标对象。处理器间消息包括了源处理器任命，而当收到处理器间消息时，每个处理器的运行中系统存储源处理器的名称和源对象的符号名。

“别名表”包含了注册的本地处理器的所有“别名”。如果一个名字没有列在别名表中，便要做一次搜索，检查目标对象是否位于连接器表中。如果答案是否定的，那么就对目标表作搜索，且如果目标处理器为已知，便向目标处理器发送消息。

在专利 US 4，901，231 中描述了在许多处理器上运行的多处理器系统。一个处理器中的用户进程可以访问其它处理器中的系统资源。当用户进程访问本地文件时，访问操作是通过用户文件表来进行的。当用户进程访问远程文件，访问操作是这样完成的：通过端口表，并经过端口表标识的虚拟信道到一个部分进程，然后通过用户文件表和部分进程的系统文件表来完成。

专利 US 5，187，790 涉及一个计算机系统，该系统有多个同时运行的进程，这些进程至少包括一个服务器进程和多个客户进程。每一进程都有表示对象访问权限的一系列标识，每一对象都具有带标识的访问检查列表，用以确定允许访问对象的处理器。

发明概述

本发明的一个主要目标是提供一个用介绍定义的那种系统，它运行时可以只用少量的需存储、维护和分布的地址信息。

本发明的另一个目标是提供一个通过介绍定义的一种系统，它可以不仅在数据库中而且在一个应用中（即在数据库中写和读的程序）接受简单的手工配置，并且在创建数据实体时不需要声明它属于哪 个处理器，也就是说这应该是预定义的。

本发明还有一个目标是提供一个用介绍来定义的那种系统，它允许灵活的分布和冗余转换，这保证了服务的可维护性和可用性。也就是说在冗余转换的情况下，必须更新的地址信息不可太多。

依据本发明的第一个方面，上述目标在分布式数据库系统中已经实现，在该系统中数据库的不同部分由一定数量的互连处理器分别控制处理。这些不同的数据库部分包括许多数据实体。对每个这样的数据实体，有关于该数据实体位于哪一处理器的全局信息，以及关于该数据实体在本处理器中位于何处的本地信息。全局信息以通用全局信息的形式位于系统中的每个处理器中，它对于每个预定义的数据实体集是特定的。

根据一个优选实施例，上述数据实体集包含了分布实体，每个实体都包含了位于某处理器中具体实体的一些实例信息，以及用以找到处理

器地址的信息.

数据实体既可以通过键值也可以通过数据实体标识来寻址，所述数据实体标识包含了关于数据实体属于哪个分布实体的信息以及标识数据实体的信息。

此数据实体标识可包括本地和全局数据实体标识，每个标识都有两个信息域。对于本地标识，其中的一个域标识本处理器，而另一个域标识数据实体。对于全局标识，一个域标识分布实体而另一个域标识数据实体。

本地信息至少包括在三张表中．第一张表含有本地数据实体标识，第二张表包含全局数据实体标识，而第三张表至少包含键值．全局信息包括在第四张表中，它包含分布实体的编号，每个这样的分布实体编号指向另一个处理器.

依据本发明的第二个方面，上述目标已通过在一个分布式数据库系统中，访问属于特定类数据实体的方法实现，而访问操作是利用了数据实体唯一键值和类的标识号．在所述数据库系统中，数据库的不同部分分别由一些互连处理器控制处理．数据库的不同部分包含了一些上文所指类型的数据实体，每个这样的数据实体有关于数据实体位于哪个处理器的全局信息，以及在本处理器数据实体位于何处的本地信息．搜索数据实体是利用键值从本处理器开始的．如果通过搜索发现此数据实体不在同一处理器中，那么处理的方法则包括以下进一步的步骤：产生一个逻辑分布实体编号，它标识位于某个处理器中被访问数据实体的一些实例信息，以及关于这个处理器的寻址信息．通过组合数据实体的类的信息和逻辑分布实体编号，产生相应的物理分布实体编号．被搜索的数据实体所处的数据库所在的处理器由物理分布实体编号来标识．消息发送到上文说的处理器，包含有关被搜索数据实体的信息，而该被搜索数据实体被利用已找到的处理器中的键值来作本地搜索．找到了的数据实体的拷贝返回给要求访问的处理器.

依照本发明第三个方面，所述目标通过在分布式数据库系统中访问属于特定数据实体类的数据实体的方法来实现．在该数据库系统中，数据库的不同部分分别由一些互连处理器来控制处理．不同的数据库部分包含一些上文所指类型的数据实体，且对每个这样的数据实体，有着关于该数据实体处于何处理器的全局信息，以及至于该数据实体在本处理

器中位于何处的本地信息．本地数据实体标识被创建，它除了包含有标识数据实体的信息外，还包含了有关分布实体的信息，分布实体又包括了位于本处理器中的数据实体类的一些实例的第一部分信息．为了试图用本地数据实体在本地处理器中找到数据实体，搜索首先从本地处理器开始。如果这一搜索结果表明数据实体不处于本处理器，则搜索方法包括以下进一步的步骤．本地数据实体标识被转变成包含分布实体第一部分信息的全局数据实体标识．它又包括了在特定处理器中数据实体特定类的一些实例的信息，数据实体属于哪个类的信息，以及对该处理器寻址的信息。全局数据实体标识也包含了标识数据实体的第二部分信息。利用包含在全局数据实体标识中的分布实体，对被搜索数据实体所处的处理器进行搜索．相关消息被送往上述的处理器，该处理器包括了全局数据实体标识．通过全局数据实体标识来对找到的处理器进行搜索．已找到的数据实体标识被返回给启动访问的处理器．

依据本发明第四个方面，所述的目标通过分布式数据库系统中的分布实体实现，在分布式数据库系统中，数据库的不同部分分别由一些互连处理器控制处理．这些不同的数据库部分包含了一些数据实体．上述的分布实体包括位于某特定处理器中数据实体特殊类型的一些实例的信息，以及这个处理器的地址信息．

依据本发明第五个方面，所阐述的目标通过分布式数据库系统的分布实体来实现，该分布式数据库系统中，数据库的不同部分由一些互连处理来控制处理．所述不同数据库部分包含了一些数据实体．所述分布实体包含一组预定义数据库体的共同信息和专有信息．

根据第四和第五方面，分布实体可形成部分数据实体标识，该标识包含分布实体的信息和标识数据实体的信息．

依据第六个方面，本发明涉及分布式数据库系统中数据实体的标识实体，在分布式数据库系统中，数据库的不同部分分别由一些互连处理处理器控制处理．这些不同的数据库部分包含了一些数据实体．上述标识实体包含有关分布实体的第一部分信息，它又包括了某一具体处理器中数据实体特定类型的一些实例信息，数据实体类型的信息，以及对处理器寻址的信息．标识实体还包括了标识数据实体的第二部分信息．

附图简述

参照附图，现将更详细地描述本项发明以及本发明的不同实施方

案。

图 1 用框图描述了分布式数据库系统，

图 2 描述了图 1 所示类型的传统系统的寻址原则，

图 3 描述依据本发明对图 1 所示类型数据库的主要寻址原则，

图 4 和图 5 分别描绘了依据本发明，在被搜索对象位于同一处理器和位于另一处理器的两种情况下如何寻址，

图 6a~6c 分别描述图 4、图 5 中所示进程中对象实体的内容，

图 7 所示依据图 4、图 5 与进程连接使用的寻址表，

图 8 描述了一个文件，在文件中详述把分布实体序列分配给不同的处理器，

图 9 所示一个文件，它为每一对象类描述分布实体的分布。

图 10 描绘了用图 8 和图 9 中的信息在一些处理器上加载分布实体的方法，

图 11 描绘了在对象与被访问进程不存在相同的处理器上的情况下，通过键对对象的访问，

图 12 是一个分布实体的物理标识的视图，

图 13 用框图表示了连接使用的表格，并描绘了分布实体编号翻译成网络地址的方法，

图 14 描绘两个查询进程，它们指向另一个处理器中的主对象，并且目的是透明地将主对象的拷贝对象传送给本处理器。

图 15 是一张图表，用来表述图 14 中的一个搜索进程，

图 16 描绘的是在图 14 中搜索进程完成了对主对象的检测后，发送对象拷贝的过程，

图 17 所示流程图概述了图 11 ~ 图 16 中所描述的搜索进程之一，

图 18 所示流程图概述了图 11 ~ 图 16 中描述的第二个搜索进程。

优选实施方案

在图 1 中用图示描绘了一个分布式数据库系统，假设该系统是面向对象的，即它的数据是作为对象来组织的。在这一连接中一个对象就是一些组织在一起的数据，它可以直接读取，也可以通过调用对象中的方法。在这一连接中，下面使用的概述是对象类或类型、特性和实例举一个对象类的例子，例如表示电话收费信息的对象。这个对象可以包含有电话号码、电话线路号等特性。这个对象的实例构成不同的收费。

这个系统包括了三个子网，分别注为 2， 4 和 6．子网 2， 4 和 6 分别包括了 4， 6 和 8 个处理器，在每个子网分别标为 8， 10 和 12．对相应的子网 2， 4 和 6，每个子网中的处理器之间分别通过链接 14，16 和 18 进行互连．子网 2，'4 和 6 之间通过链接 20 和 22 进行互连．

在分布式数据库系统中包括的每一个处理器可能包括在图 1 中没有示出的整个数据库的相应的恰当部分包含有一些对象，这些对象可能被其它的处理器访问．传统上，与每个对象相连接的除了有关于子网和对象所处处理器的部分信息，有关于在另一处理器中执行所需服务的代理的部分信息，以及该对象在处理器内存中的确切位置的信息，刚才提到的关于在另一处理器中代理的部分信息在某些情况下又称为“通信进程”。与当前例程有关的服务在处理器内存中获取对象，并处理本处理器和另一处理器之间进行的通信，这将在下面作进一步的描述．

如果包含有这些部分信息的全部信息将存在于所有处理器上，那必将产生一个非常大的寻址表，并且在整个数据库系统上对对象地址作必要的大量更新操作。这在图 2 中作了描述，图 2 显示了部分数据库系统，也就是子网 2．图中四个处理器 8 中的每一个都包含一张寻址表 24，对所有的处理器都是相同的．每张寻址表 24 中都有本处理器所有对象的指针以及数据库系统中其余处理器中对象的指针．用箭头 26 表示，指向某些对象 28．为清楚起见，右图中处理器的一些箭头被删掉了，每当产生一个新的对象，它的地址就必须分布给所有别的处理器，或者记录在一个中央目录中，在访问的时候可以从那里获得地址信息．这对于对象的分布是非常灵活的，但却意味着庞大的寻址表。

图 2 中还有一个指向一张寻址表 24 的箭头 30，用来表示一个对象的引入，该对象识别上述表格中的键值并完成对数据库系统中任何对象的访问。

简言之，本发明是建立在这样一个思想上： i.a 表明对象位于哪个处理器上的全局信息，对于大多的对象是共用的．正如下面还要详细描述的那样，这个全局信息对所有的处理器都是同样的，为了清楚起见，它以地址表的形式显示在图 3 中，并只在一个处理器的 32 处标出。

全局信息 32 包括有关一些分布实体的信息，每个实体都包含有某一对象类的一些实例信息。

图 3 中，这些分布实体中的某一些标为 34．对每个这样的分布实体

34，表 32 中都有它属于哪一处理器的描述．在图 2 的连接中提到的键值于 36 处被转换为与某一分布实体 34 相应的索引．对于每个分布实体 34，表 32 将包含一个地址指针，指向该分布实体所属的处理器．在图 3 的 38， 40， 42 和 44 处分别标注了一些这样的地址指针．更加具体的是，这些指针指向位于每个处理器的表 46，从表 46 出发，下一个地址指针 48 指向相应子数据库中位于分布实体 34 的对象．一些这样的对象用点 50 标出．

参照以下的图表，将更详细地阐述依据本发明的寻址原则的实施方案。

从应用层来看，一条数据库对象记录可以用两种方法来寻址．即用键的方法或者用数据对象标识．

图 4 中，一个用户进程标为 52，一个处理器标为 54．进程 52 用键值的方法访问处理器 54 内存中对象 58 的属性 56．这个访问用箭头 60 标出，它是由代理对象 62 中的响应方法来完成的．代理对象 62 是为用户进程 52 中的目的而产生出来的．作为响应，对象 58 返回箭头 64，一个存在特性 56 中的本地对象标识，并指向处理器 54 中另一个对象 66．参照用箭头 68 表示．进程 22 用本地对象标识来打开对象 66，并且用在对象 66 中读取的属性来产生第二个包含响应方法的代理对象 70，用箭头 72 表示．在图 4 中，对象标识就是这样用来作同一处理器中对象寻址的。

在图 4 中通过对象 58 迂回到达对象 66，是预先假定从一开始就已知可以找到在对象 58 中对对象 66 的参照．而且这只是一个例子，在面向对象数据库中，数据通常是以这样的方式来建模：在数据库中，访问需要“导向”，也就是说，仿照参考从一个对象到另一个对象。

图 5 中情况与图 4 不同，被寻址的对象位于另一个处理器 74，标为 66′．在这个例子中，在处理器 54 的内存里产生了一个对象 66′的拷贝 66″．如箭头 72′标出，用户进程 52 通过代理对象 70 访问拷贝 66″中的属性．这将在后面作更详细的描述。

从上文中讲述的来看，对象标识可以被应用用来作本地对象或位于另一处理器对象的寻址．如果象图 5 中那样对象并不位于本处理器，那么对另一处理器的分布式访问将以对应用完全透明的方式处理完成，这叫做隐式分布。对象标识通过数据库分布逻辑被转换成全局对象标识。

对象标识也可以被应用程序在自己的分布协议中使用，所谓开放式分布，不过首先要由应用程序将其转换作外部使用，尤其是对于全局对象标识。

参看图 6a，上文所讨论的类型的对象标识应包括两个域，例如每个都是 32bits．第一个域表示对象所属的分布实体，另一个域表示标识对象的序列号．分布实体的标识由数据库系统外的支持系统产生，而序列号是与一个应用系统中的对象几乎同时地被指定的，即在数据库中的读写程序．对象的外部可见的名称与分布实体之间的连接是在设计阶段就决定了的，并且在应用系统的整个生命周期都不再改变．

依据图 4 中所举例，参看图 6b 对象标识在第一个域中有值，比如说是 0，表示对象，即 66，位于本处理器，因此称为“本地对象标识”．在图 5 所示的情况下，并参看图 6c，对象标识如上文所述，被转换成一个全局对象标识．正如下面将作的更详细的描述，这种转换是通过给对象标识的第一个域中赋分布实体的值来实现的，这个值是用来查找出对象所属的处理器．

参考图 7 以及上文对图 3 ~ 5 的描述，每个处理器分别包含四张表：79、80、82 和 84．

表 79 是用于对象识别键值．表 79 是在图 4 所示的情况下由进程 52 来找到对象 58，方法是使用了表中与当前键值有关的指针．虽然没有标注出来，但实际上在每个处理器中，每个设置了键值的对象类都可有一张键值表 79．

表 80 包括了指向同一处理器中对象的本地对象标识，因此叫做“本地处理器”．在图 4 的例子中，数据库处理器找到了第一个区域中的零，于是进入表 80，并找到对象 66．

表 82 包括两列，86 和 88，第一列 86 存分布实体号，而第二列 88 通常指向另一个处理器，或者逻辑上指向那个处理器的数据库处理器．在本例中，假设第二列是指向一种通信端口，在美国专利 08/193,844 中有描述，在瑞典专利 9300431 - 5 中有相应描述，并且它与第二个处理器中进行的活动有关．这里引入美国专利申请 08/193,844 以备参考．依据上述美国申请，通信端口（为简单起见，下文中称为端口），将表示属于操作系统通信机制的类型资源，并且有一个活动用它来建立连接．这里使用的活动这个概念，在同一个美国专利申请 08/193,844 中也被使

用，用来将操作系统中产生的作业链，定义为：独立的外部或内部事件加上在作业链执行过程中使用的资源总数．这里作业的进一步含义，还是在美国专利申请中解释为：一种指向进程的表象（ phenomenon ），使得进程拥有的对象中的方法被执行，该连接中的作业能够产生指向其它进程或本进程的新作业．

表 84 相当于图 3 中的表 46，它包含两列： 84′和 84″，第一列用来作全局对象标识，而另一列用作处理器内存中相应的对象的指针．

根据上面的解释，在下面对图 5 所述例子的描述中，为清楚起见，“端口”这个概念将由相关概念“处理器”或“数据库处理器”代替．

依据图 5 中例子，在本处理器 54 中的数据库处理器进入这一处理器的表 82，并依据箭头 89 找到处理器 74．消息送往另一处理器 74 中的数据库处理器，该消息包含一个指示器，指出是否需要通过键或全局对象标识来搜索该对象．如图 5 中假设，如果全局对象标识是已知的，通过进入在处理器 74 的表 84 中的全局对象标识全局对象标识的列 84′．处理器 74 中的数据库处理器继续搜索进程，依据箭头 90．表 84 中，与找到的全局对象标识在同一行，但在另一列 84′中，依据箭头 92，找到对象 66′的指针．

反过来，如果上述指示器表示搜索应通过键值来进行，正如没有全局对象标识的情况下一样，那么处理器 74 中的数据库处理器，依据箭头 94，将搜索进程指向处理器 74 的表 79．依据箭头 95，利用被搜索对象 93 的键值，这一对象的地址便可找到．

如果在图 5 的例子中，处理器 74 的内存里有属于同一分布实体的另外的对象，也就是说，是具有相同的分布实体号和不同的序列号，那么该进程被传送给处理器 74．

参照图 8 － 18，在这里将阐述两个实际的实施方案，其中的表 79，80， 82， 84 将被更详细地描绘出来．

下面的对象类描述中列举了一个用描述对象类语言，比如 DELOS，写的固有（ persistont ）对象类，这在 Gote Andersson 发表在电信杂志 NO.4/93 上名为“开发软件”的文章中提到．这里特别提到类 Subscriber 的对象的问题，它包含了两个属性，其中一个是主键，即一个分布属性．

```
OBJECT TYPE Subscriber IS
PERSISTENT PROPERTIES
PRIMARY KEY subscrNum;
--MDP-sequence 0..logicalMDPhigh
ATTRIBUTES
subscrNum: SubscriberNumber;
iAge: Integer
END;
```

在这例说明中，第一行定义对象类，即 **Subscriber**．下一行描述存储特性。随后的是对属性名称 **PRIMARY KEY**（主键）的描述，即 **SubScr Num**．

在定义… **MOP-Sequence O … logieal MDPhigh** 中，**MDP**（主数据部分）代表分布实体．这个定义是有关所讨论的对象类的最大分布率的信息，即对象类分布实体的最大数量．该信息用来给配置步骤产生一个输入数据，下面会作详细阐述，并且允许系统能够在运行中检查，函数 **KeyToMDP** 并不返回声明范围之外的值，在下文中会作更详细的描述．

在 **ATTRIBUTES**（属性）下的第一行，说明 **SubScr Num** 是 **Subscriber Number** 类型的，在第二行说明 **SubScr Num** 中名为 **iAge** 的方法函数是 **Integer**(整型）的．这两条属性都可以预定义类型．

说明的最后一行进一步用“**IS STRING**”（为字符串）来定义属性类型 **Subscriber Number** 的特性，即属性表述为数字串．

从上面对象类的描述以及下面有关对象类方法的类似描述，用特定的数据定义语言通过编译器生成该语言的代码．以上的叙述及下面叙述的例子也是建立在使用代理对象的基础上的，代理对象的类型在美国专利申请 **08/264,059** 中有阐述，相应的在瑞典专利 **93021756** 也有有关叙述，依据发明，它是在使用寻址方法的情况下用来完成数据访问的．刚才提到的美国专利申请被引入以备参考之用．在这项美国专利申请中也阐述了用本文描述的方法来产生代码，因此这里就不需作进一步说明了．在该美国专利中，代理对象被称为 **DOA**，代表“数据对象代理”，在这里有时也会使用这个名称．

在用来将键值转换为分布实体的方法中，方法的结构用 **DELOS** 生成，但应用程序设计者必须写明键值是如何翻译成分布实体号的．它的算法形成 **DOA** 类的一部分并依赖于被寻址的对象类．在本例中假设主键

= 电话号码 1111122．则所说的方法就通过以下来执行:

DisosDbMDP

SubScriber:KeyToMdp(Key)

其中 KeyToMDP 称为转换函数，下面会作进一步阐述.

本例中假设最后两位的数字 22 被一个 00 ~ 99 之间的值所屏蔽．这个屏蔽后的值就成为分布实体号 22 .

方法要被创建（实例）操作和打开（实例）操作使用，也就是说分别为实例的产生和打开.

为访问对象，即在对象中读或写，对象的一个本地拷贝被安装在正在运行的处理器中，参看图 5 中的 66″，以及如上文图 5 的阐述．在这个连接中，参照被制成两种类型的拷贝，即“惰性（ lazy ）”拷贝与“活性（ agile ）”拷贝．一直使用的缺省是“惰性”拷贝，意思是通过访问，获取对象．“活性”拷贝是预先配制好的并且作活性响应的单元是整个分布实体.

为安装配置，应对所需处理器作主数据分配及活性拷贝．对每个分布实体都完成这一工作，方法是说明一系列主或活性分布实体应分别分配给一个处理器或一个处理器池，即在分布式数据库系统中的多处理器系统．在图 8 所示文件中对此有具体描绘.

在图 8 中,第一列表示主或活性 - MDP .第二列表示一个物理 MDP 序列,第二列的两个子列表示对象类和 MDP 号.例如在第一行中， 9914 是对象类号码，标识该对象类， 0 和 49 表示分布实体号，也就是说，它表示一个类号码为 9914 的对象有五十个分布实体.最后一列表示应该安装主或活性 - 分布实体的处理器.

此外，当向一个应该载入系统的加载模块输入数据时，还需要每个对象类分布实体的分布信息．这里的加载模块的概述与上面提到的美国专利申请 08/264,059 中的是相同的．所说的分布实体的分布在图 9 所示文件中列举，该文件从图 8 的文件中获值.

图 9 中第一列表示对象类名称（对象类型），第二列表示对象类编号（ dbClassId ）．第三列显示对象类的分布实体编号(logical MDP high)，分布实体编号从 0 开始．这两个对象类对于可能产生的对象的编号没有什么限制，不过在表格中说明了分布实体号，即对象可以分布的最大的处理器数是， Subscriber 为 100， Lic 为 10 .

图 8 和图 9 包括用于所有对象类的用分布实体分配信息作的配置数据。这类信息应加载在本地数据库的所有处理器中。具体地，分布实体分配信息是在数据加载阶段，用数据库中的初始化函数由数据库向类对象加载的。因此它包含在基本功能中。

图 10 中描绘了装载过程的物理上的表现。为简单起见，只显示图 8 所述处理中的两个处理器，即图 8 第 1 和第 3 行的处理器 4 与处理器 2 。

更具体地，图中显示对象类 Subscriber 的分布实体 100 是如何载入处理器 4 $^{\#}$ 和 2 $^{\#}$ 中的，这两个处理器分别包含在子数据库 102 和 104 中。序列 0→49 中的主分布实体 106 被载入处理器 4 $^{\#}$ 的内存中，而在序列 0→33 中，活性分布实体 108 被载入处理器 2 $^{\#}$ 的内存中。

以上定义的对象类用以下程序载入数据库中:

Void Subscriber::install("1111122")

Object instance（对象实例），例，由 **Sbux = Sbuscriber：： create**（"1111122"）产生。

上面所提将属于分布实体 22 的对象实例，它依据对处理器的分布实体分配，在处理器 4 $^{\#}$ 的内存中初始化。据上文所述，一个键值通过某个算法转换成分布实体编号。这个算法是一个主数，输入参数是主键数据类型，并返回一个在 0 到 **logical MDP high** 之间的整数。这一函数将可能的键值平均地分配给被标识的分布实体，不过在对象类设计者考虑自己的设计时，可以考虑选择一种合适的算法。

对象被打开可以通过下式来更新:

Subx=subscriber::open Update ("1111122")

数据库可以通过数据库控制处理器来对应用作透明的分布式访问的控制处理，这就是前文中提到过的隐式分布。

在分布式数据库系统中对一个对象的搜索首先在本地处理器开始，利用图 7 中的表 79 或表 80 去找到当前主对象，它取决于搜索利用键值还是本地对象标识来完成。

图 11 更加详细地描述了，如果对象与访问进程不处于同一处理器时，是如何用键值来访问对象的。更具体地，这一进程在 110 处表示在另一处理器中执行什么。箭头 112 表示它搜索处理器 118 上子数据库 116 中的本地键值表 114，表 114 相当于图 7 中的键值表 79 。数据库 116 的数据库控制处理器用上面所述的 DOA 方法 **Subscriber::keyToMoP**

（key）来调用转换函数 keyToMdp（key），如箭头 120 所示，它用键值作为参数，得到一个逻辑 MDP - NO.. 依照图 12，通过合成实例的类编号和逻辑 MDP 号，可以得到物理 MDP - NO.. 这个物理 MDP - NO 构成图 6c 中全局对象标识的第一个域.

分布实体所属的处理器被在图 7 中的表 82 里查找. 然后消息被送往对象所在的处理器或者是这个处理器中由数据库处理进程公布（pbulish）的端口. 因为分布实体编号不能唯一标识对象，消息中包含对象类和主键. 为了在需要的时候可以设读锁或写锁，消息还说明了将要对对象作何种操作.

分布实体标识是用来找到网络地址的，这里利用了分布在分布式数据库系统中的所有处理器上的数据库内部表. 这些表在图 13 中的框图表示，没有要求与实际相符. 在实际的实现中，这些表更加紧凑.

根据图 13，在每个处理器中这些表包括一张搜索表 130 和第二张与图 7 中表 82 相当的表. 搜索表 130 中，每一对象类有相应的一行，用来说明对象类的编号. 表 82 中每个安装了的分布实体有相应一行，每行分为三列. 每行的第一列表示对象类编号，第二列是分布实体号，而第三列是列对象的端口名称. 更具体地，从第二张表可见，对对象类编号 myclass id 来讲，有 102 个分布实体，它们具有到对象的端口名的相关标识.

知道所搜索对象的类编号后，在本处理器中的分布处理控制器在表 130 中开始搜索，箭头 132 所示，经过箭头 134 表示的搜索进程，直到找到对象类编号 myclass id . 依据箭头 136，将顺序找出表 82 中一组编号为 myclass id 的对象类，这里假设用箭头 138 表示的搜索找出了编号为 100 的分布实体. 依据箭头 140，最后的结果是对被搜索对象所处的当前处理器进行寻址.

在图 14 和 16 中，对透明分布作更详细的描述，参考图 7， 11 和 13，在那里曾对此有过简单叙述.

图 14 中的开始状态是，先假设分布开始的处理器中的分布处理控制器执行了一个搜索进程，该进程可以是使用全局对象标识，也可以是使用键值的，参看图 13，对此有描述. 这些在箭头 142 处用 MDP 100 + mykey 表示.

在标号 150 的处理器中，由包含处理器 150 的数据库 156 中分布处

理器（标号为 154 ）产生一个接口代理 152 . 消息被打包成输出形式，并发送给标号为 160 的另一个处理器，见箭头 158 . 该处理器在相应的子数据库 162 中，包含了被搜索的主控对象. 消息到达数据库 162 后，被接口代理 168 接收. 接口代理 168 由它的分布处理器 166 产生. 当控制处理处理器间的分布式通信的通信进程被激活后，消息将被拆包，并且对对象进行本地搜索.

如果消息中含有全局对象标识，搜索通过数据库 162 的表 174 来执行，如箭头 170 和 172 . 表 174 相当于图 7 中的表 84 ，箭头 170 ， 172 和图 7 中的箭头 90 表示相同的过程. 被找到的对象在 175 处表示出.

在图 15 中用表格描述了在被搜索对象的全局标识未知的情况下，在处理器 160 中利用对象类编号和主键来进行的另一种搜索进程，象图 13 中的例子那样，图 15 中用框图显示的这些表的使用只是原理性的，不要求与实际情况相符. 在实际的应用中，这里的表也是更紧凑的.

依据图 15 的表包含在每个处理器中，它们包括一张搜索表 180 和另一张与图 7 中表 79 相当的表格. 表 180 中每个对象类对应一行，在这行表示对象类的编号. 表 79 中每个安装了的键对应一行，每行分为三列. 每行的第一列表示对象类编号，第二列表示键，第三列包含了一个指向数据库对象的指针.

见图 15 所示，处理器 4 # 扣的数据库控制处理器利用被搜索对象的对象类编号 myclass Id ，执行一个搜索进程，见箭头 182 ，该进程从表 180 开始，通过箭头 184 ，找到所说的对象类编号 My Class Id ，然后通过箭头 186 ，得到一组编号为 My Class Id 的对象. 最后，通过按箭头 188 搜索，找到主键和相应的指向被搜索对象的指针 190 .

图 15 中的表格在图 14 中描绘为块 79 .

图 14 中的起始状态是，假设子数据库 156 中的分布处理器通过表 130 ， 82 执行了一个与上文所述图 13 中类似的搜索过程. 分布是从子数据库 156 开始. 如前文所述，在本处理器 150 中由它的子数据库 156 中的分布处理器 154 产生一个接口代理 152 . 消息被打包成输出形式，然后发送给另一处理器 160 ，见箭头 158 ，数据库 162 包含了被搜索主控对象 192 .

当消息到达子数据库 162 ，它被接口代理 168 接收. 当处理控制处理器间分布式通信的通信进程已被激活，消息便被拆包，并且通过数据

库 162 的表 182 和 79，执行对对象的本地搜索，参看图 15 中所描述的搜索进程.

当主控对象 175 或 192 被找到后，在其上加写操作锁. 这个写操作锁必须从处理器 160 上的数据库处理器获得，处理器 160 包含主控对象，因为在另一个处理器上，锁的分配是由数据库中分布处理控制器自动执行的. 与此同时，参考图 16，惰性拷贝对象 200 将被返回到数据库 156，参见箭头 202 . 数据库 156 包含处理器 150，搜索事件在该处初始化，参看图 5 中 66″. 处理器 150 中的 204 表示开始访问的进程.

为清楚起见，上文参考图 11 ~ 16，对透明分布的进程描述，用流程图的形式概述，并表示在图 17 和 18 中.

图 17 中，利用类编号和键值开始对对象的搜索，步骤 250 .

搜索先在本处理器中进行，去找当前主对象. 这个过程利用键值，在本地键值表 79 中进行，参看图 7 . 如果在步骤 252 中，发现对象位于同一个处理器，那么进程依据箭头 253，转向图 17 中的最后步骤，下面还将进一步说明.

如果在步骤 252 中发现，对象不能在同一处理器中找到，依据图 11 的有关描述，在步骤 254 中通过转换函数 keyToMDP(key)得到逻辑分布实体，在步骤 256 中，通过合成对象类编号和逻辑分布实体得到物理分布实体，参见图 12 .

然后，在步骤 258，对处理器数据库进程的端口标识进行搜索，被搜索对象在处理器的子数据库中. 这些都是用图 13 中的表格来搜索完成的.

步骤 260 中，消息被发往端口. 由于分布实体号不足以唯一地标识对象，消息中包括了类编号和键值. 为了在需要时设置读锁或写锁，消息中还表示了将对对象作何种操作.

在图 14 通过在被找到处理器中的表 79 中作本地搜索，找到被搜索对象后，在步骤 262，一个拷贝对象被送往开始访问的处理器，并安装在那里，参考图 16 有关描述.

最后，在步骤 264，产生一个数据库代理 DOA，并返回应用进程，参见美国专利申请 08/264,059 中有关描述.

在图 18 中，利用对象标识开始对对象的搜索，步骤 266 .

搜索先在本处理器中进行，通过利用本地对象标识和本地表 80，去

找到当前主对象，参看图 7．如果在步骤 268 中，发现对象位于本地，那么进程依据箭头 269 直接继续到图 18 中的最后步骤。

如果在步骤 268 中发现，对象不在同一处理器中，应用程序就将本地对象标识转换为全局对象标识，步骤 270．参看以上有关图 5 的描述。

然后，在步骤 272，对处理器中数据库进程的端口标识进行搜索，被搜索对象在处理器的子数据库中．这是利用图 7 中表 82 的分布实体号，通过搜索来完成的．分布实体号包含在全局对象标识中。

在步骤 274 中，消息被送往端口．消息中包含了全局对象标识．为了在需要时可以加读锁或写锁，消息还应该可以表明将要对对象作何种操作。

在图 14 里，通过在被找到处理器的表 174 中作本地搜索，找到被搜索对象后，在步骤 276，一个拷贝对象被送往开始访问的处理器，并装在那里，可参看图 16 有关描述。

最后，在步骤 278，产生一个数据库对象代理 DOA，并返回应用进程。

说 明 书 附 图

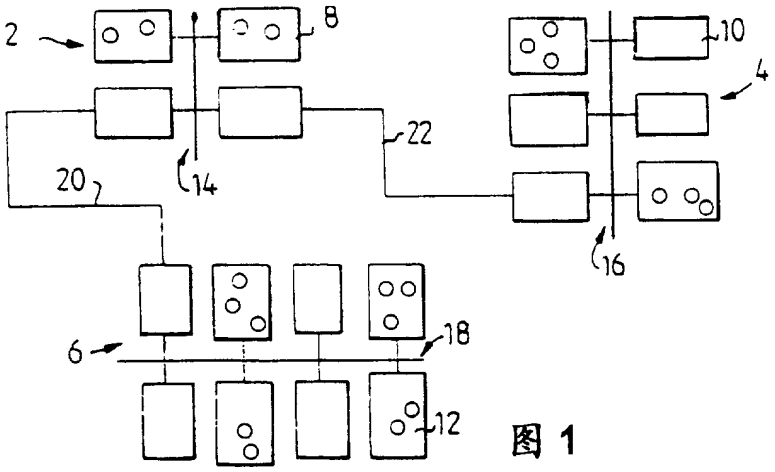


图 1

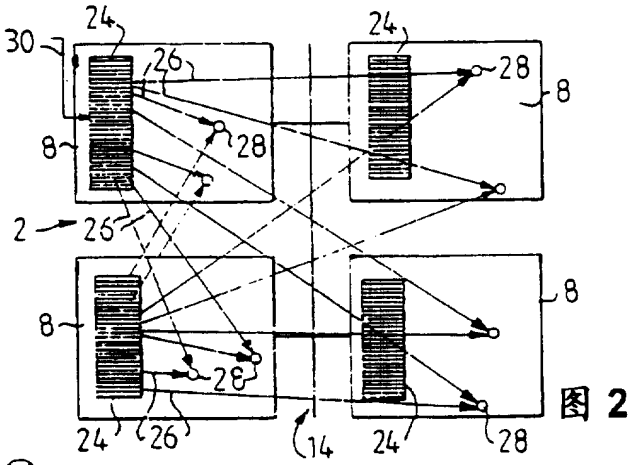


图 2

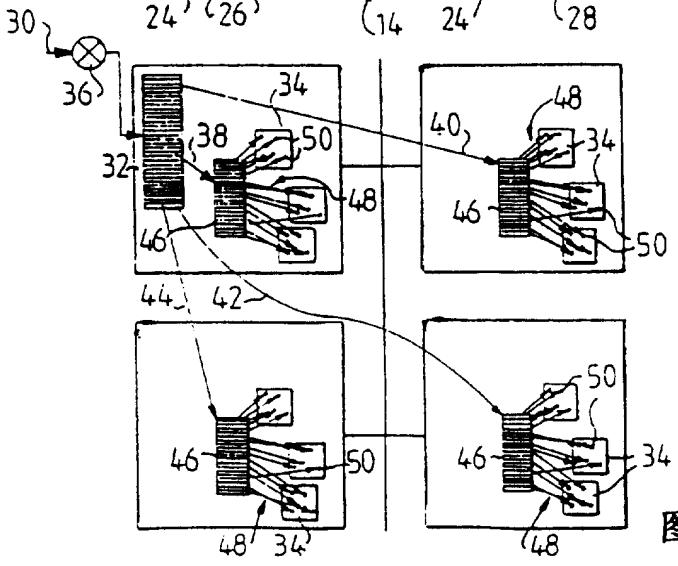


图 3

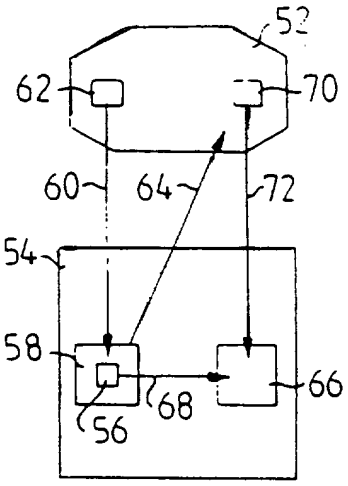

52
62
70
60
64
72
54
58
66
68
56

图 4

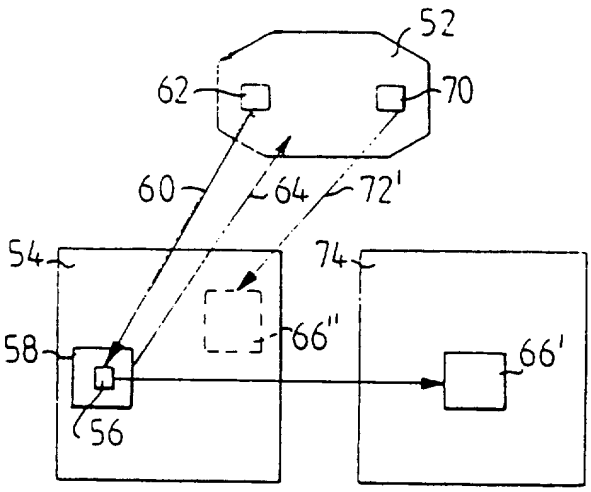

52
62
70
60
64
72'
54
74
66''
58
66'
56

图 5

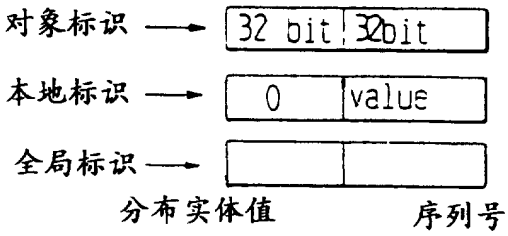

对象标识
32 bit
32bit
本地标识
0
value
全局标识
分布实体值
序列号

图 6

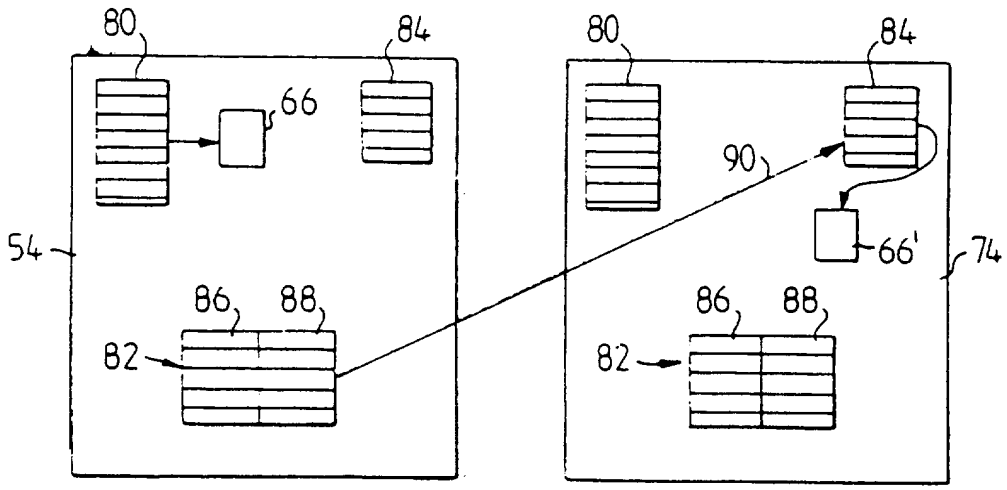

80
84
66
54
86
88
82
80
84
90
66'
74
86
88
82

图 7

主 /可变MDP	物理MDP序列	处理器
主	<9914,0> ..<9914,49>	处理器 4
主	<9914,50>..<9914,99>	处理器 5
可变	<9914,0> ..<9914,33>	处理器 2
可变	<9914,34>..<9914,66>	处理器 6
可变	<9914,67>..<9914,99>	处理器 8
主	<7122,0> ..<7122,9>	处理器 1

图 8

对象名	类型	对象类型号	MDP: S数
Subscriber		9914	99
Lic		7122	9

图 9

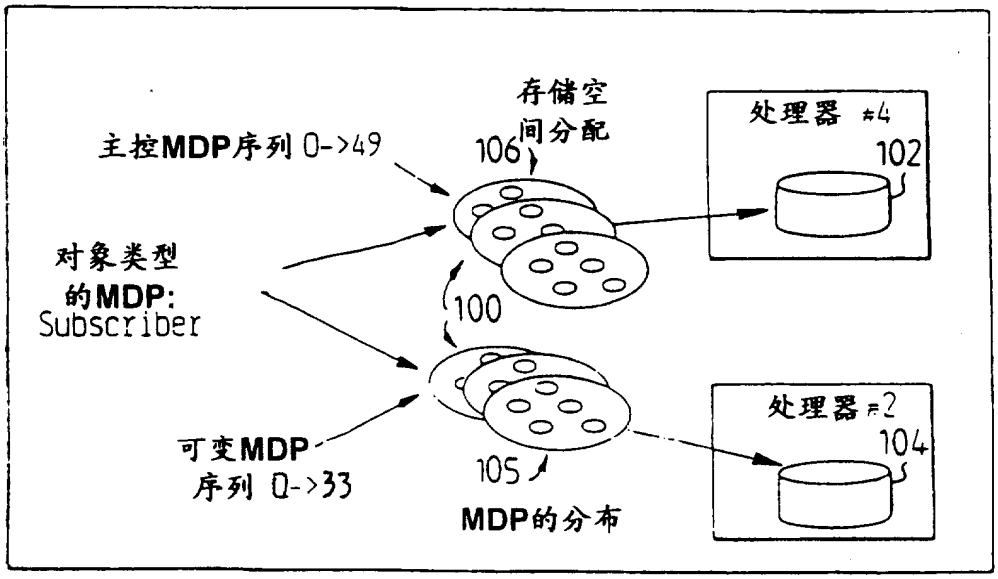


图 10

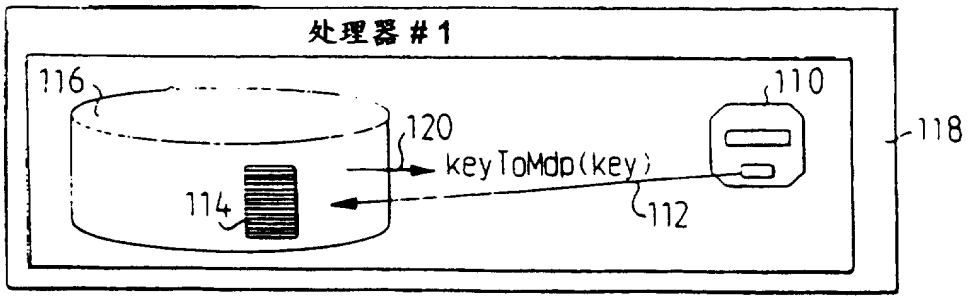

处理器#1
116
110
120
118
keyToMdp(key)
114
112

图 11

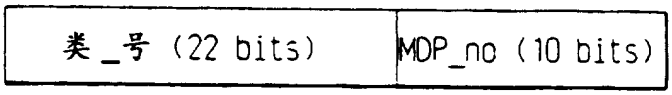

类_号 (22 bits)
MDP_no (10 bits)

图 12

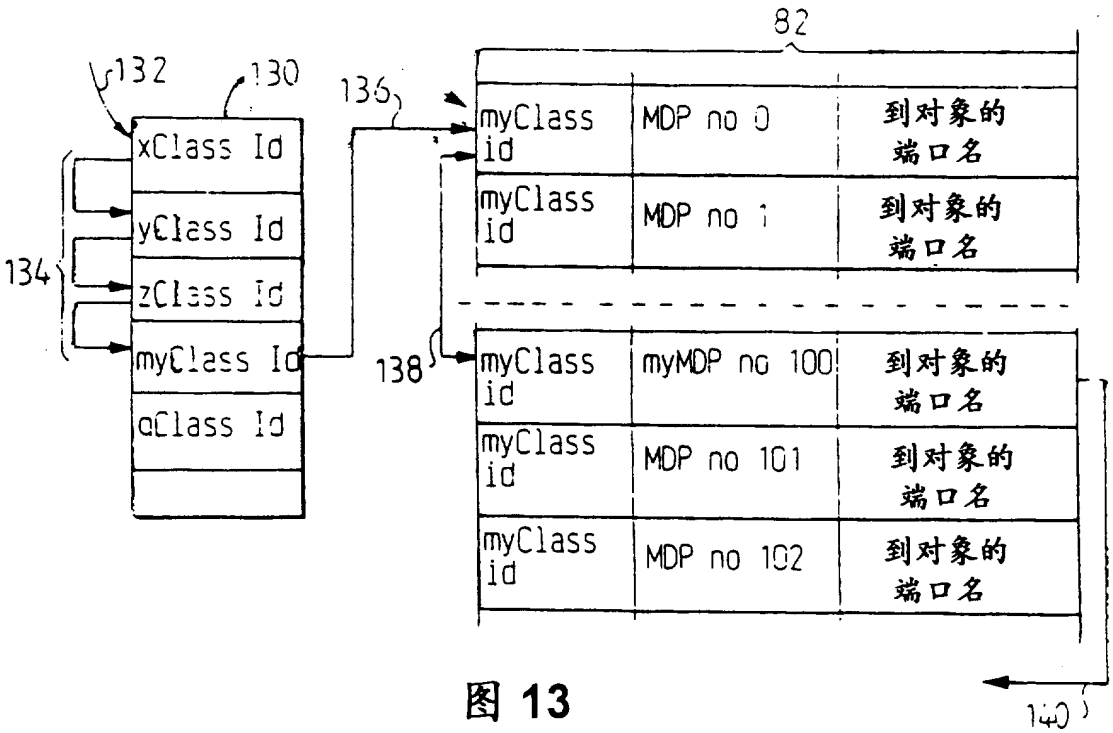

82
132
130
136
134
138
140
xClass Id
yClass Id
zClass Id
myClass Id
aClass Id
myClass id
MDP no 0
到对象的端口名
myClass id
MDP no 1
到对象的端口名
myClass id
myMDP no 100
到对象的端口名
myClass id
MDP no 101
到对象的端口名
myClass id
MDP no 102
到对象的端口名

图 13

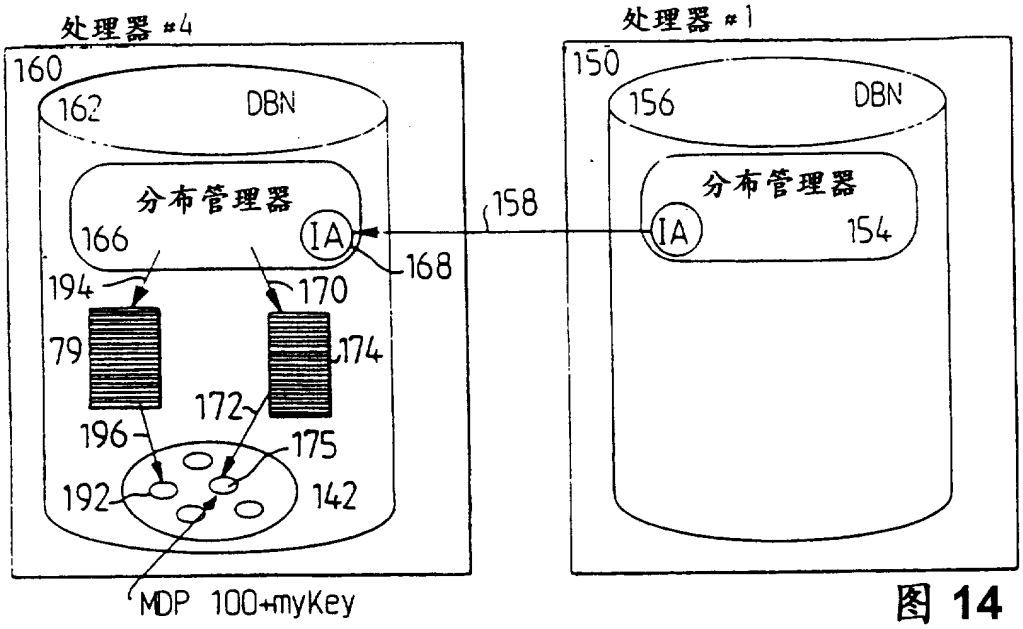

处理器 #4
160
162
DBN
分布管理器
166
IA
168
158
处理器 #1
150
156
DBN
分布管理器
IA
154
194
170
79
174
172
196
175
192
142
MDP 100+myKey

图 14

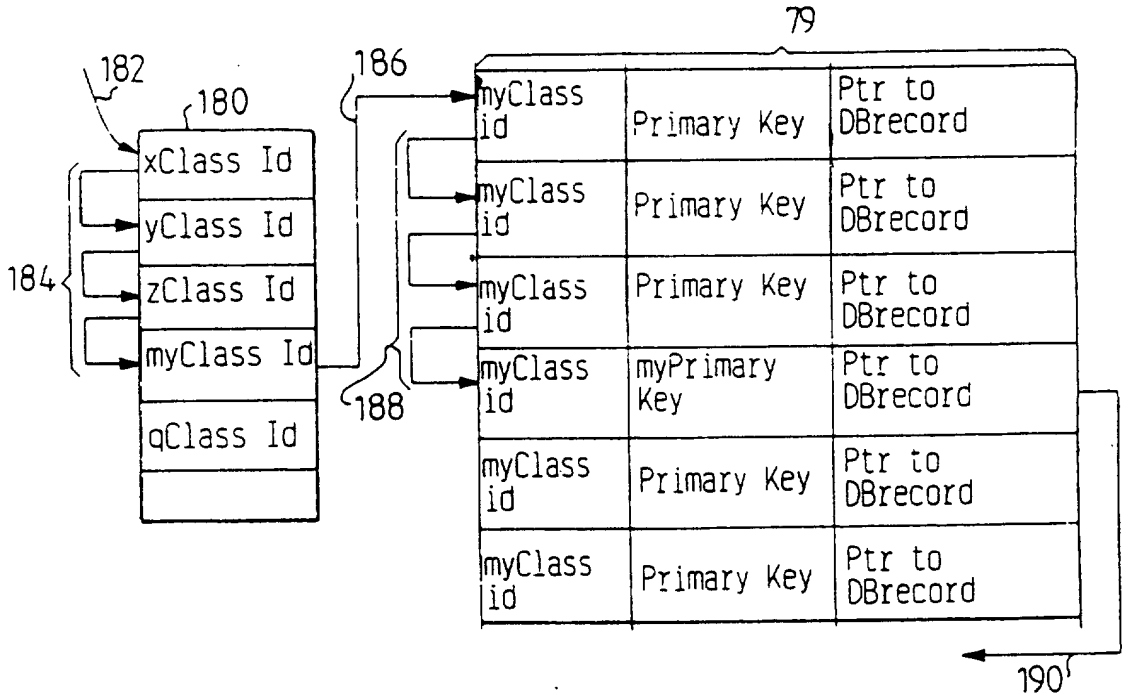

79
182
180
186
xClass Id
yClass Id
zClass Id
myClass Id
qClass Id
184
188
myClass id
Primary Key
Ptr to DBrecord
myPrimary Key
190

图 15

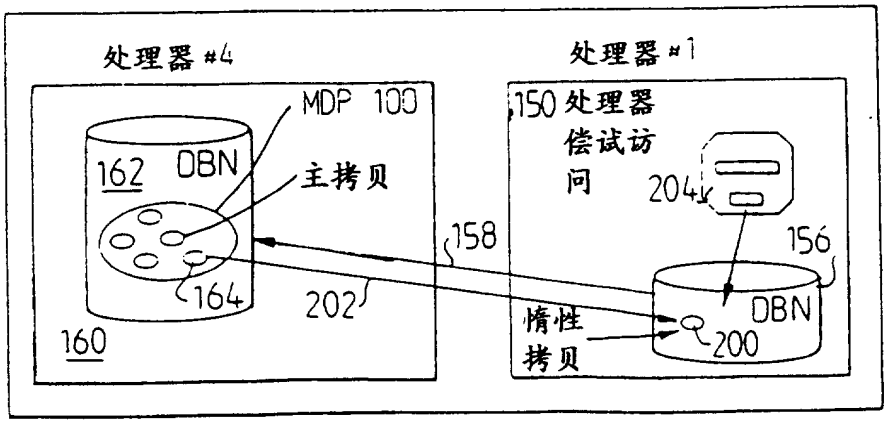

处理器 #4
MDP 100
162
DBN
主拷贝
164
160
158
202
处理器 #1
150 处理器
偿试访
问
204
156
DBN
200
惰性
拷贝

图 16

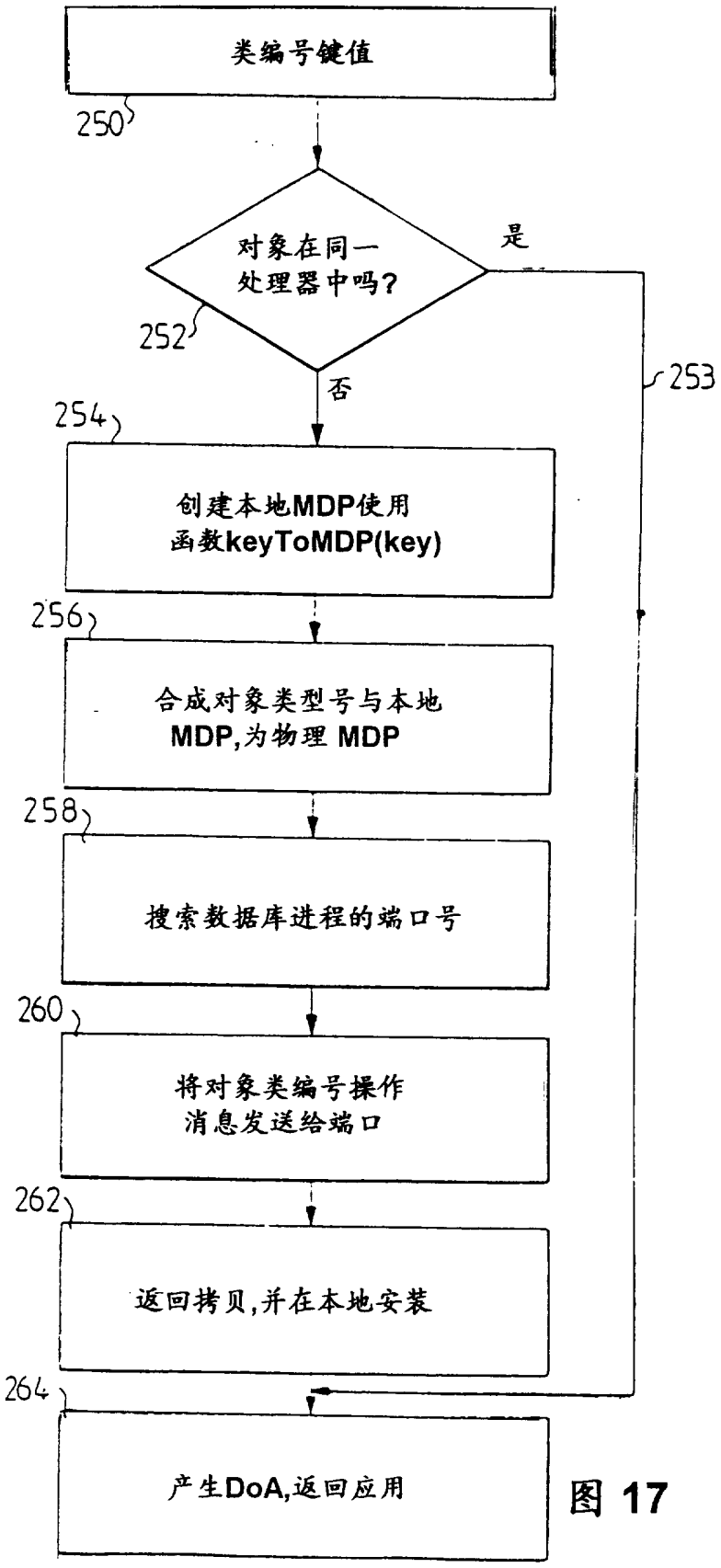

类编号键值
250
对象在同一
处理器中吗?
是
252
否
253
254
创建本地MDP使用
函数keyToMDP(key)
256
合成对象类型号与本地
MDP,为物理 MDP
258
搜索数据库进程的端口号
260
将对象类编号操作
消息发送给端口
262
返回拷贝,并在本地安装
264
产生DoA,返回应用

图 17

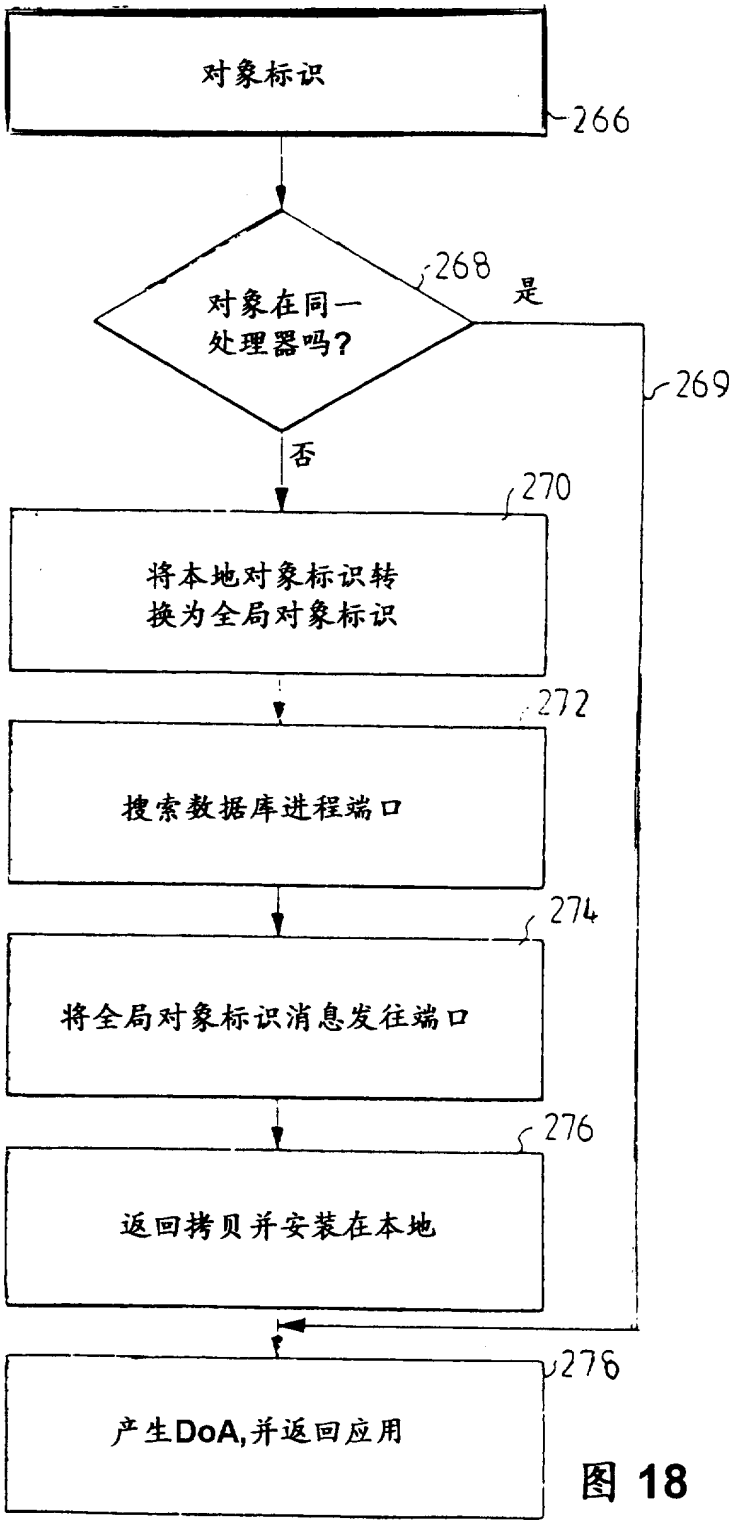

对象标识
266
对象在同一处理器吗?
268
是
269
否
270
将本地对象标识转换为全局对象标识
272
搜索数据库进程端口
274
将全局对象标识消息发往端口
276
返回拷贝并安装在本地
276
产生DoA,并返回应用

图 18